Scalable Multigrid Methods Using Fortran90 and MPI

Dennis Parsons and Ali Namazifard

Department of Civil Engineering

University of Illinois

December 1, 1999

Center for Simulation of Advanced Rockets

University of Illinois at Urbana-Champaign

Outline

- Multigrid algorithms for implicit dynamics
- Element level parallelism
- Nested mesh generation and partitioning
- Algorithm performance
- Application to ASCI coupled solid rocket simulations
- Extensions to adaptive mesh refinement
- Future developments

Nonlinear Implicit Dynamics

Loop over time:

Loop over Newton iterations:

Solve

$$\left(\frac{1}{\beta\Delta t^2}\boldsymbol{M}+\boldsymbol{K}_T^{(k)}\right)\Delta\boldsymbol{u}^{(k)}=\boldsymbol{f}_{t+\Delta t}-\boldsymbol{i}_{t+\Delta t}^{(k)}-\frac{1}{\beta\Delta t^2}\boldsymbol{M}\boldsymbol{u}_{t+\Delta t}^{(k)}+\frac{1}{\beta\Delta t^2}\boldsymbol{M}\boldsymbol{u}_t+\frac{1}{\beta\Delta t}\boldsymbol{M}\dot{\boldsymbol{u}}_t+\boldsymbol{M}\ddot{\boldsymbol{u}}_t$$

Increment displacements

$$\boldsymbol{u}_{t+\Delta t}^{(k+1)}=\boldsymbol{u}_{t+\Delta t}^{(k)}+\Delta\boldsymbol{u}^{(k)}$$

End loop over Newton iterations

Compute velocities and accelerations

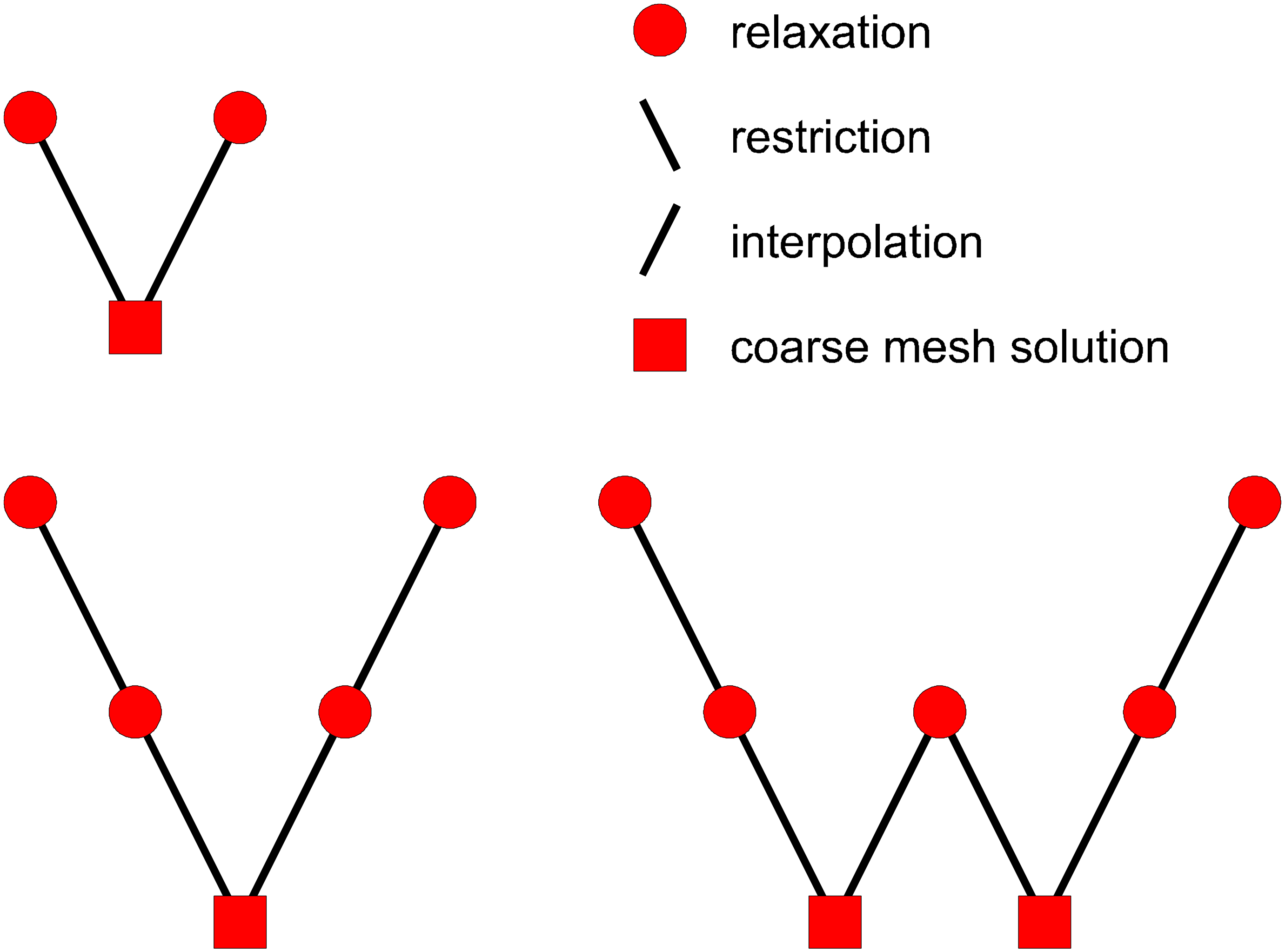
$$\ddot{\boldsymbol{u}}_{t+\Delta t}=\frac{1}{\beta\Delta t^2}\left(\boldsymbol{u}_{t+\Delta t}-\boldsymbol{u}_t\right)-\frac{1}{\beta\Delta t}\dot{\boldsymbol{u}}_t-\ddot{\boldsymbol{u}}_t$$

$$\dot{\boldsymbol{u}}_{t+\Delta t}=\dot{\boldsymbol{u}}_t+\frac{\Delta t}{2}\left(\ddot{\boldsymbol{u}}_t+\ddot{\boldsymbol{u}}_{t+\Delta t}\right)$$

End loop over time

Parallel Multigrid Methods

- Use multigrid to solve linear matrix equations in nonlinear algorithms
- Basic two grid method
  - iterative methods quickly produce a smooth error on a fine mesh
  - compute the smooth error on a coarse mesh
- Recursion produces multigrid method
- Computational effort is linearly proportional to problem size (algorithmically scalable)
- Element level parallelism gives scalable performance


relaxation
restriction
interpolation
coarse mesh solution

Multigrid Components

- **Relaxation**
  - **Jacobi, Gauss-Seidel perform poorly for ill-conditioned problems**
  - **preconditioned conjugate gradient**
- **Interpolation, restriction**
  - **nodal averaging for nested meshes**
- **Coarse mesh solution**
  - **compute coarse mesh matrix from coarse mesh elements**
  - **preconditioned conjugate gradient**

PCG Relaxation

Given $\quad \boldsymbol{x}^{(0)}, \; l = 0$

Initialize
$$\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}$$
$$\boldsymbol{d}^{(0)} = \boldsymbol{A}_D^{-1}\boldsymbol{r}^{(0)}$$
$$\boldsymbol{p}^{(0)} = \boldsymbol{d}^{(0)}$$

Iterate
$$\alpha^{(l)} = \frac{\left\langle \boldsymbol{r}^{(l)}, \boldsymbol{d}^{(l)} \right\rangle}{\left\langle \boldsymbol{p}^{(l)}, \boldsymbol{A}\boldsymbol{p}^{(l)} \right\rangle}$$
$$\boldsymbol{x}^{(l+1)} = \boldsymbol{x}^{(l)} + \alpha^{(l)}\boldsymbol{p}^{(l)}$$
$$\boldsymbol{r}^{(l+1)} = \boldsymbol{r}^{(l)} - \alpha^{(l)}\boldsymbol{A}\boldsymbol{p}^{(l)}$$
$$\boldsymbol{d}^{(l+1)} = \boldsymbol{A}_D^{-1}\boldsymbol{r}^{(l+1)}$$
$$\beta^{(l+1)} = \frac{\left\langle \boldsymbol{r}^{(l+1)}, \boldsymbol{d}^{(l+1)} \right\rangle}{\left\langle \boldsymbol{r}^{(l)}, \boldsymbol{d}^{(l)} \right\rangle}$$
$$\boldsymbol{p}^{(l+1)} = \boldsymbol{d}^{(l+1)} + \beta^{(l+1)}\boldsymbol{p}^{(l)}$$
$$l = l + 1$$

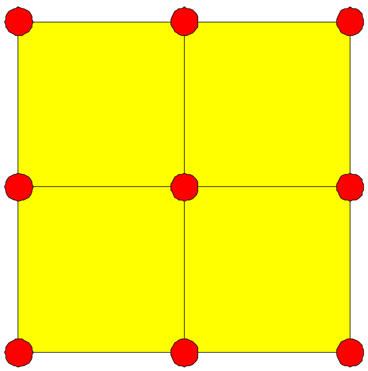
Primary operations

- matrix-vector multiplications $\boldsymbol{A}\boldsymbol{p}^{(l)}$
- DAXPYs $\quad \boldsymbol{p}^{(l+1)} = \boldsymbol{d}^{(l+1)} + \beta^{(l+1)}\boldsymbol{p}^{(l)}$
- scalar products $\left\langle \boldsymbol{p}^{(l)}, \boldsymbol{A}\boldsymbol{p}^{(l)} \right\rangle$
- preconditioning $\quad \boldsymbol{d}^{(l+1)} = \boldsymbol{A}_D^{-1}\boldsymbol{r}^{(l+1)}$

Intergrid Transfer Operators

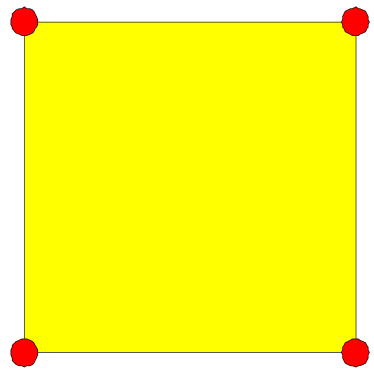
Fine mesh

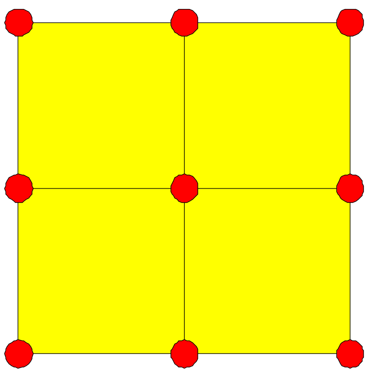
Coarse mesh



$$\Delta \boldsymbol{x}_f^{(k)} = \boldsymbol{I}_c^f \Delta \boldsymbol{x}_c^{(k)}$$

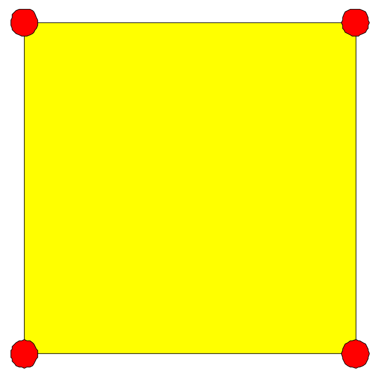
$$\boldsymbol{I}_c^f = \boldsymbol{T}$$





$$\boldsymbol{r}_c^{(k)} = \boldsymbol{I}_f^c \boldsymbol{r}_f^{(k)}$$

$$\boldsymbol{I}_f^c = \boldsymbol{T}^T$$



Primary operations

matrix-vector multiplications

$$\boldsymbol{r}_c^{(k)} = \boldsymbol{T}^T \boldsymbol{r}_f^{(k)}$$

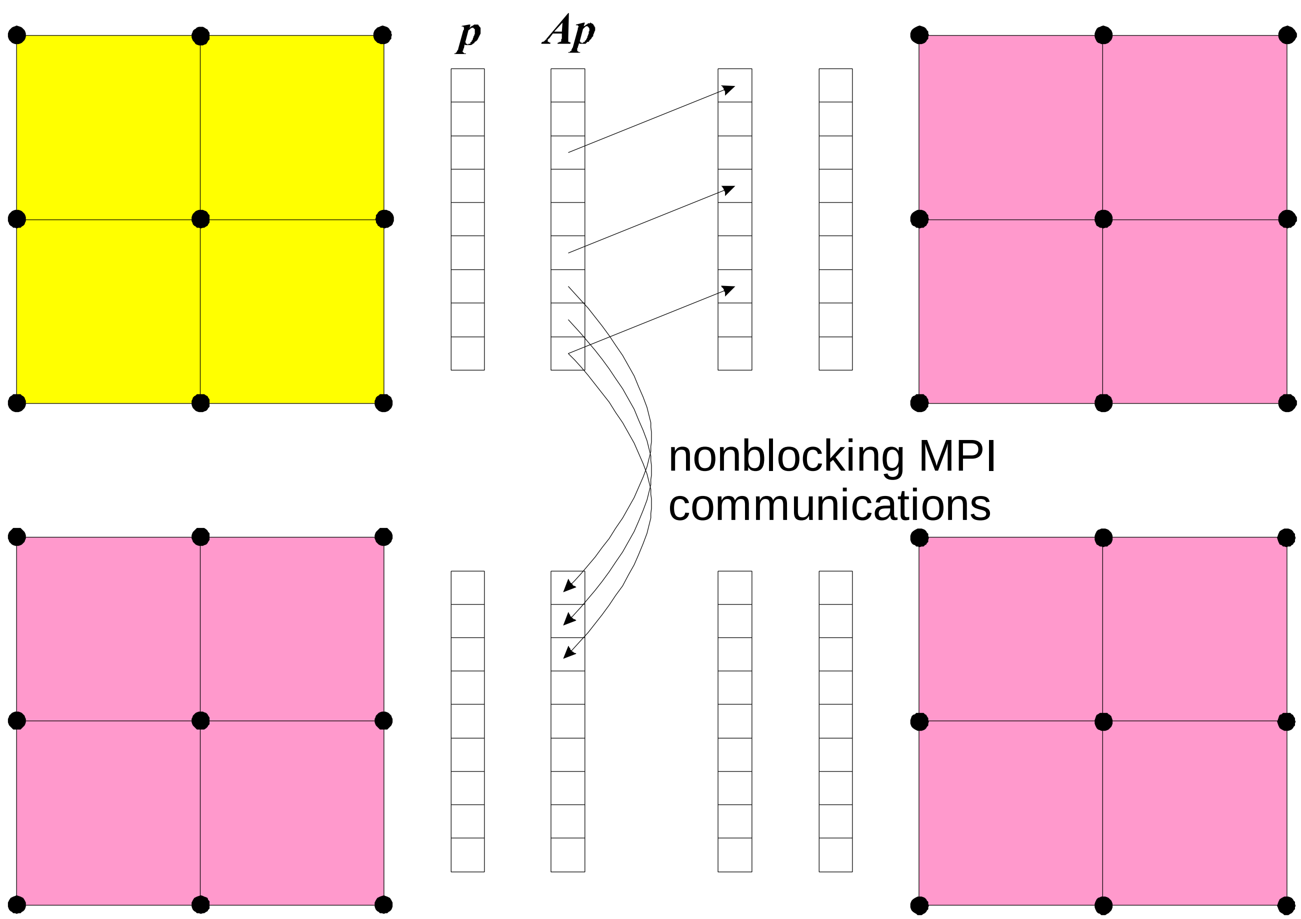
$$\Delta \boldsymbol{x}_f^{(k)} = \boldsymbol{T} \Delta \boldsymbol{x}_c^{(k)}$$

Element Level Computations

- **All operations can be performed independently on partitioned domains**
- **Interprocessor communications required during**
  - **matrix-vector multiplications**
  - **scalar products**
  - **fine to coarse mesh restriction**
- **Matrix-free computations reduce storage and CPU time**

$$\boldsymbol{K}\boldsymbol{p} = \sum_e \boldsymbol{K}^e \boldsymbol{p}^e = \sum_e \left(\int_{\Re^e} \boldsymbol{B}^T \boldsymbol{D}\boldsymbol{B} \, dV \right) \boldsymbol{p}^e$$

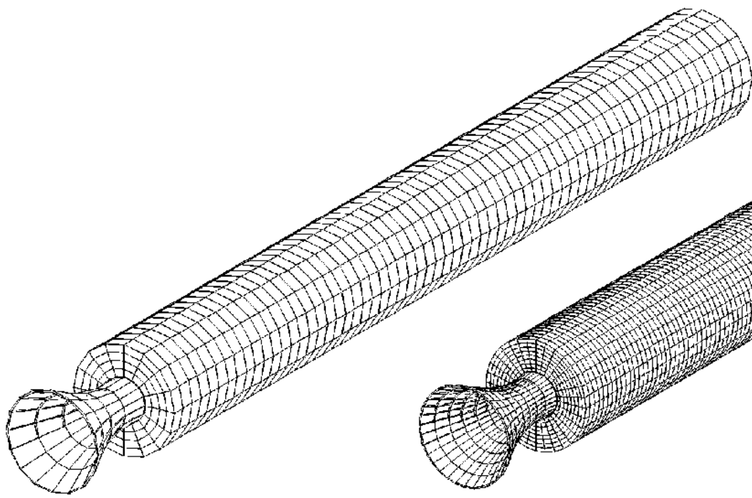
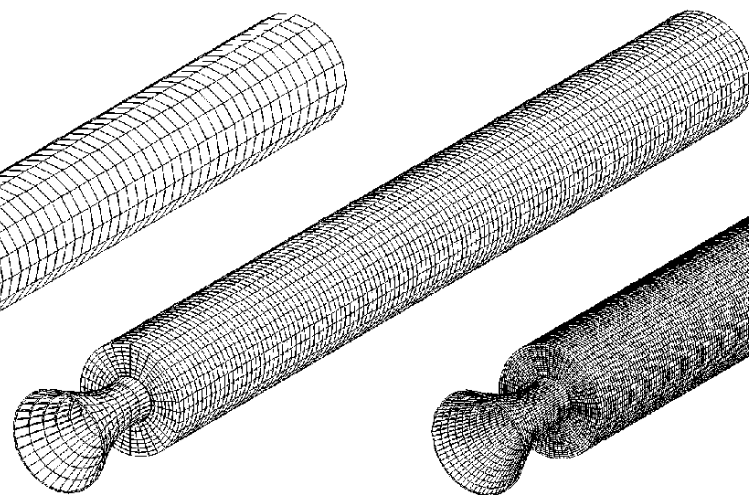
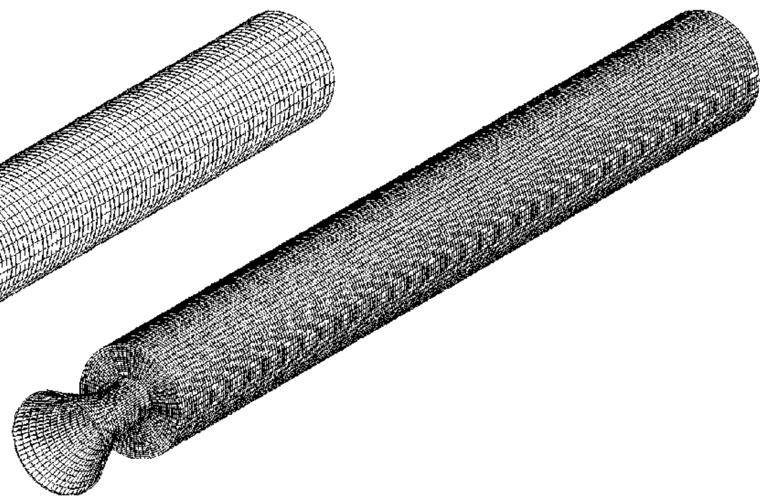
Distributed Memory Implementation



Mesh Generation

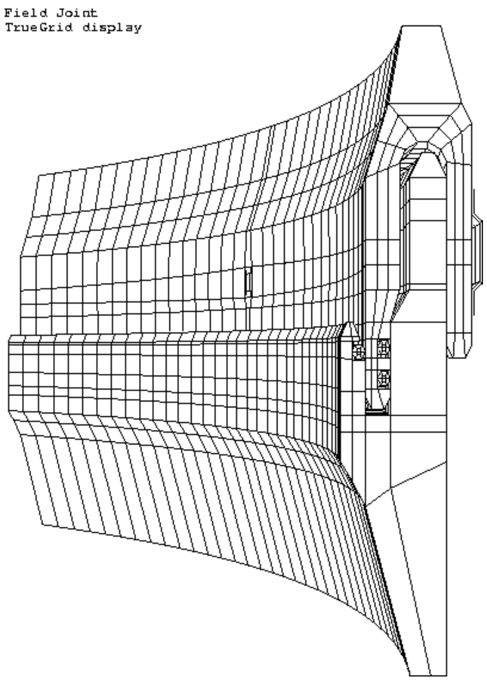
- Multigrid requires a hierarchy of increasingly finer meshes
- Adaptive mesh refinement will eventually be used to generate this hierarchy
- *Truegrid* is employed to produce a sequence of nested, uniformly refined hexahedral meshes
- Complex parts can be modeled in this manner

Solid Rocket Motor

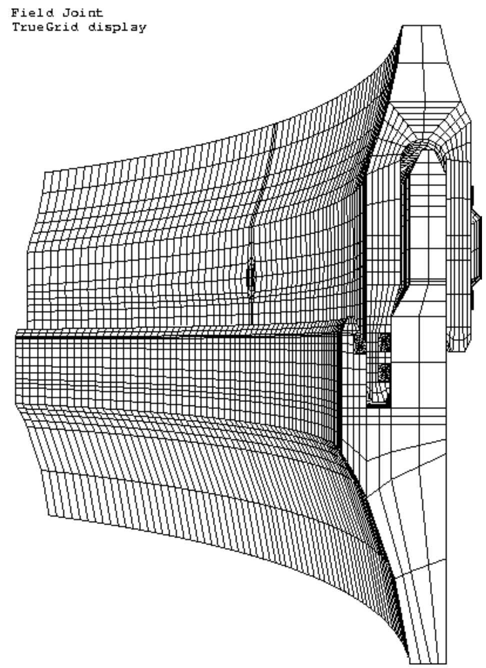




4,096 32,768 262,144

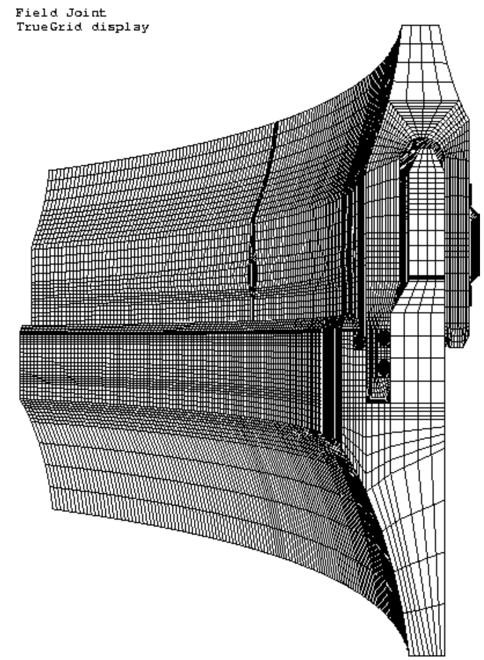
Rocket Joint Detail



3,624



28,992

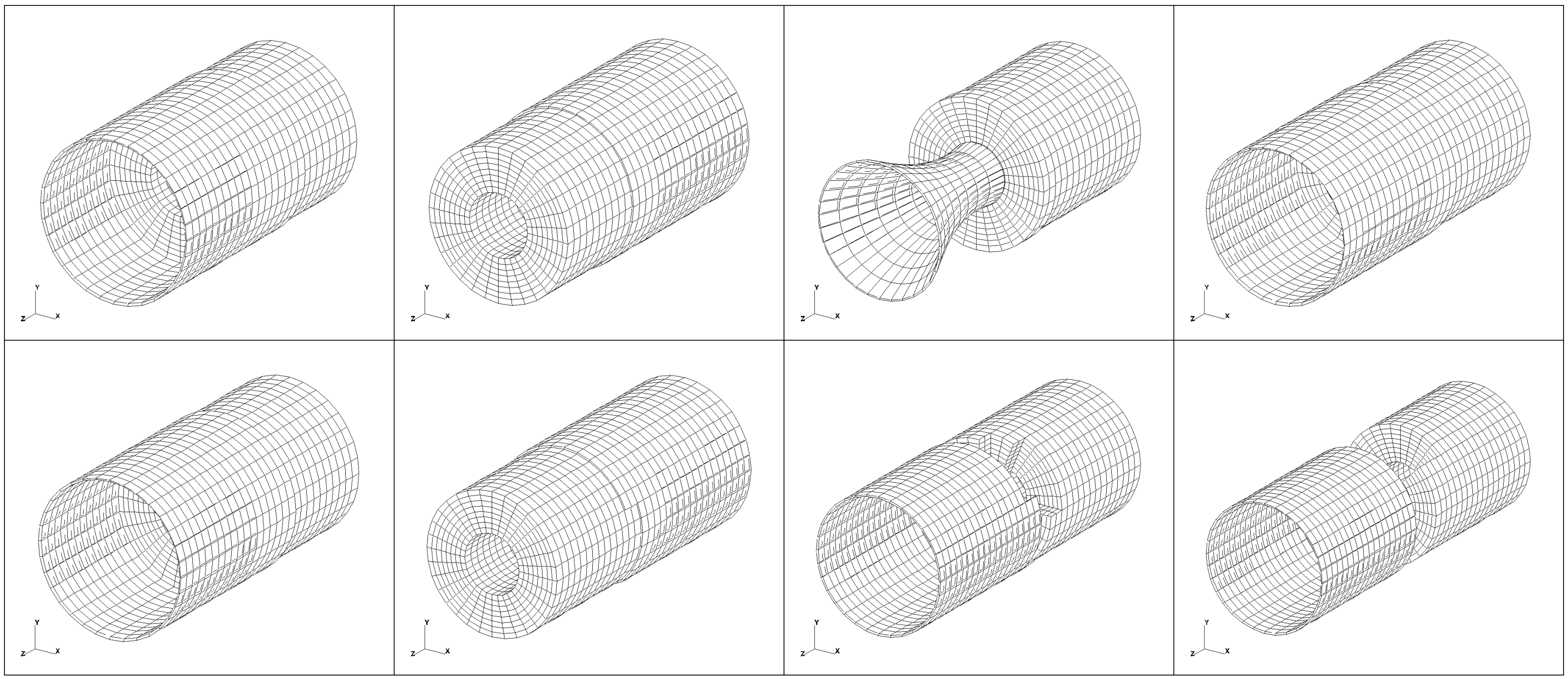


231,936

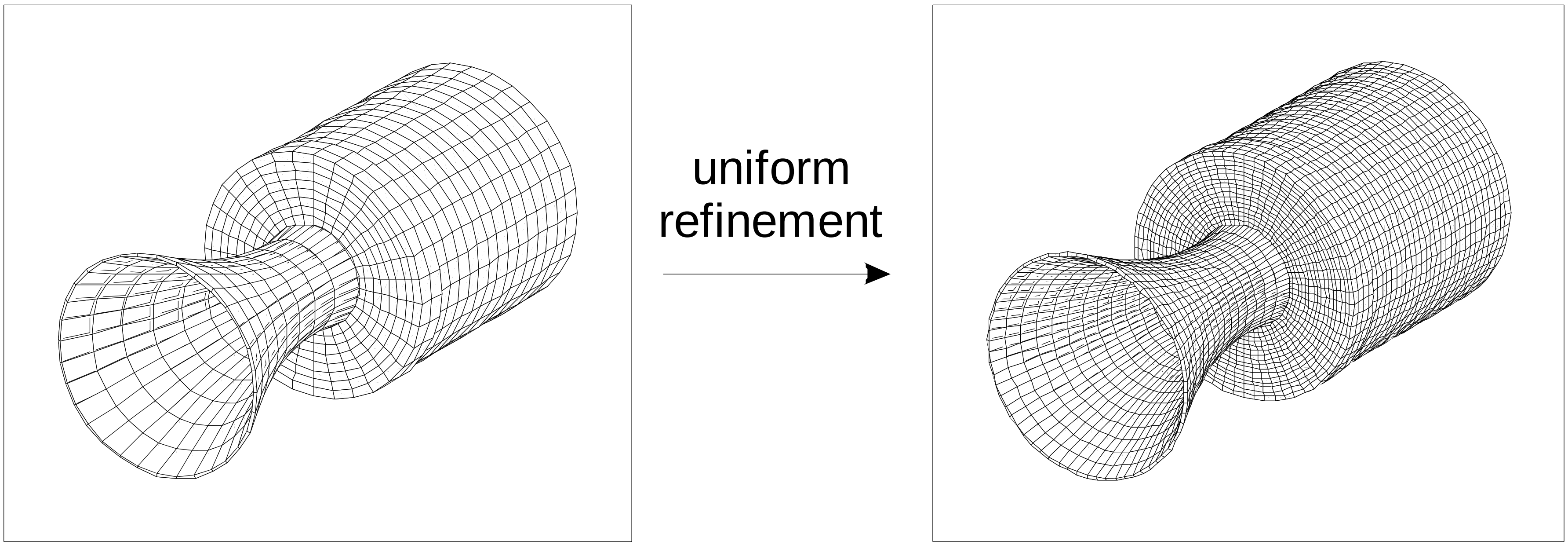
Mesh Partitioning

- Partitioning is performed on the coarsest mesh using *Metis* to achieve perfect load balance
- Uniform refinement of the coarsest mesh partitions produces partitions on all of the fine meshes
- Perfect element load balance is maintained throughout the mesh hierarchy
- Communications may not be optimum

Coarsest Mesh Partitions



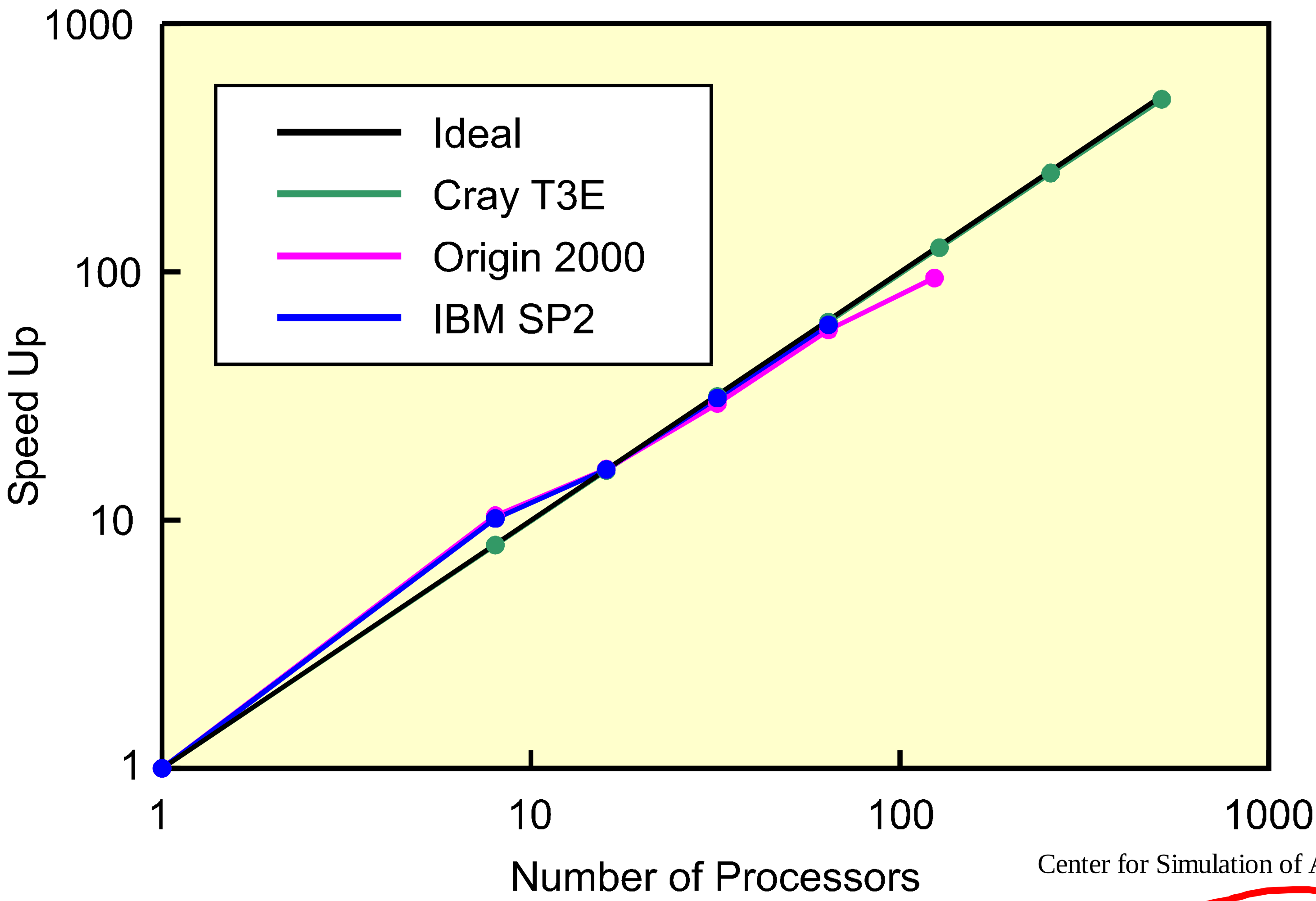
Partition Mesh Refinement



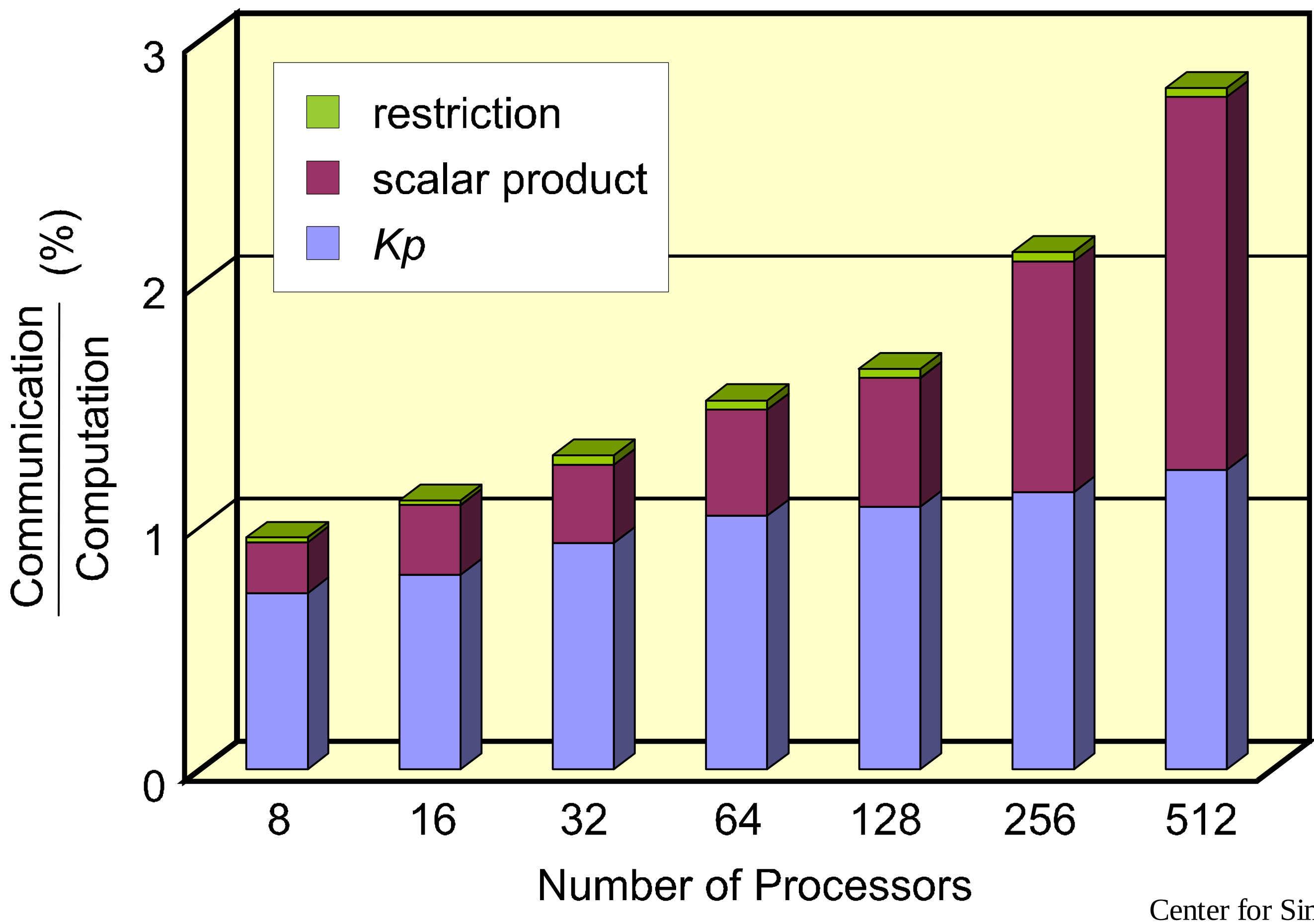
Parallel Performance

- Code has been benchmarked on several multiprocessor machines
  - IBM SP2 (Argonne)
  - SGI Origin 2000 (NCSA)
  - SGI Cray T3E (PSC)
- Computation dominates communication
- Scalable element computations
- Cray T3E showed the best performance
- Lazy processors on Origin 2000

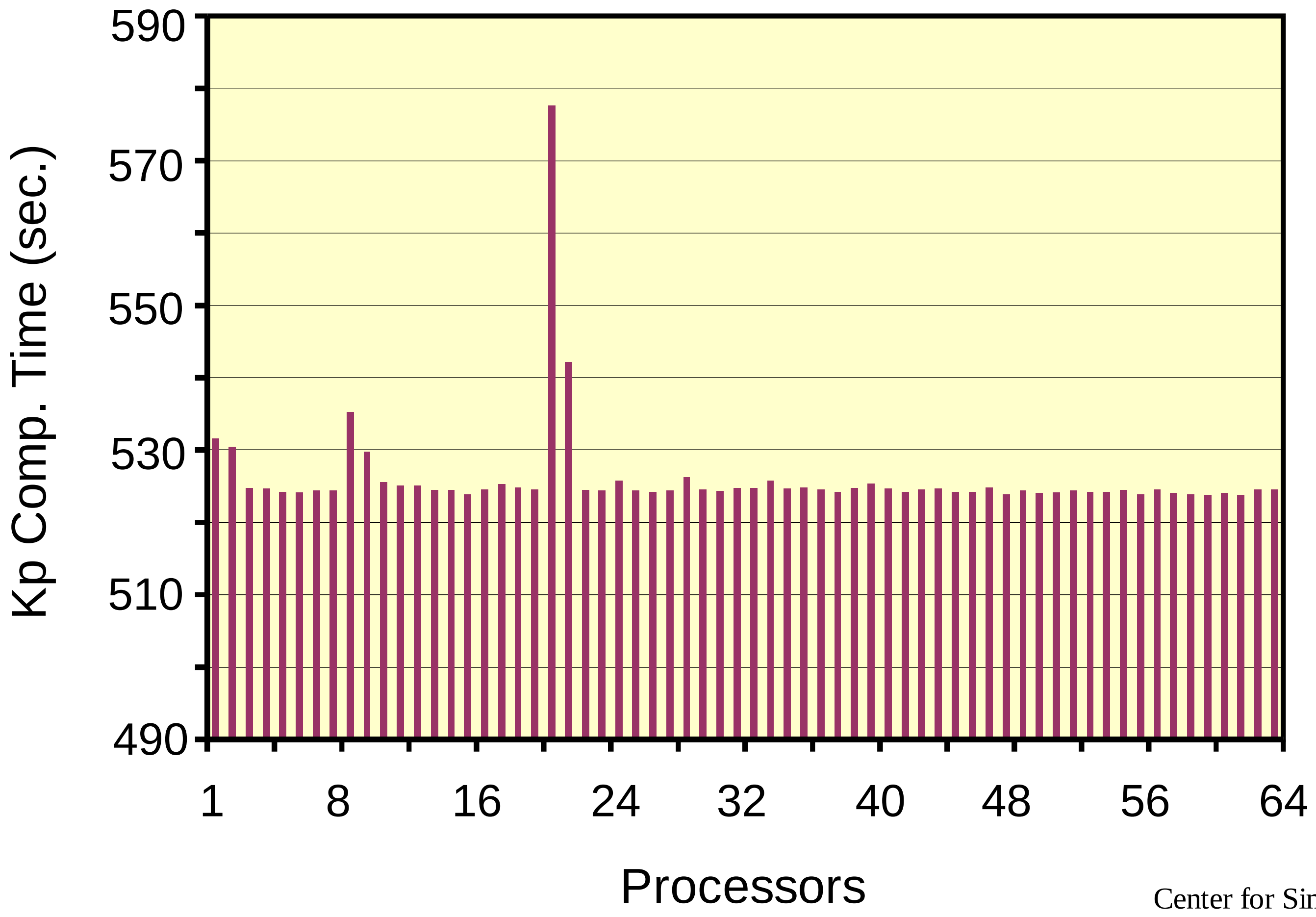
Scalability Results



Cray T3E Cost Analysis



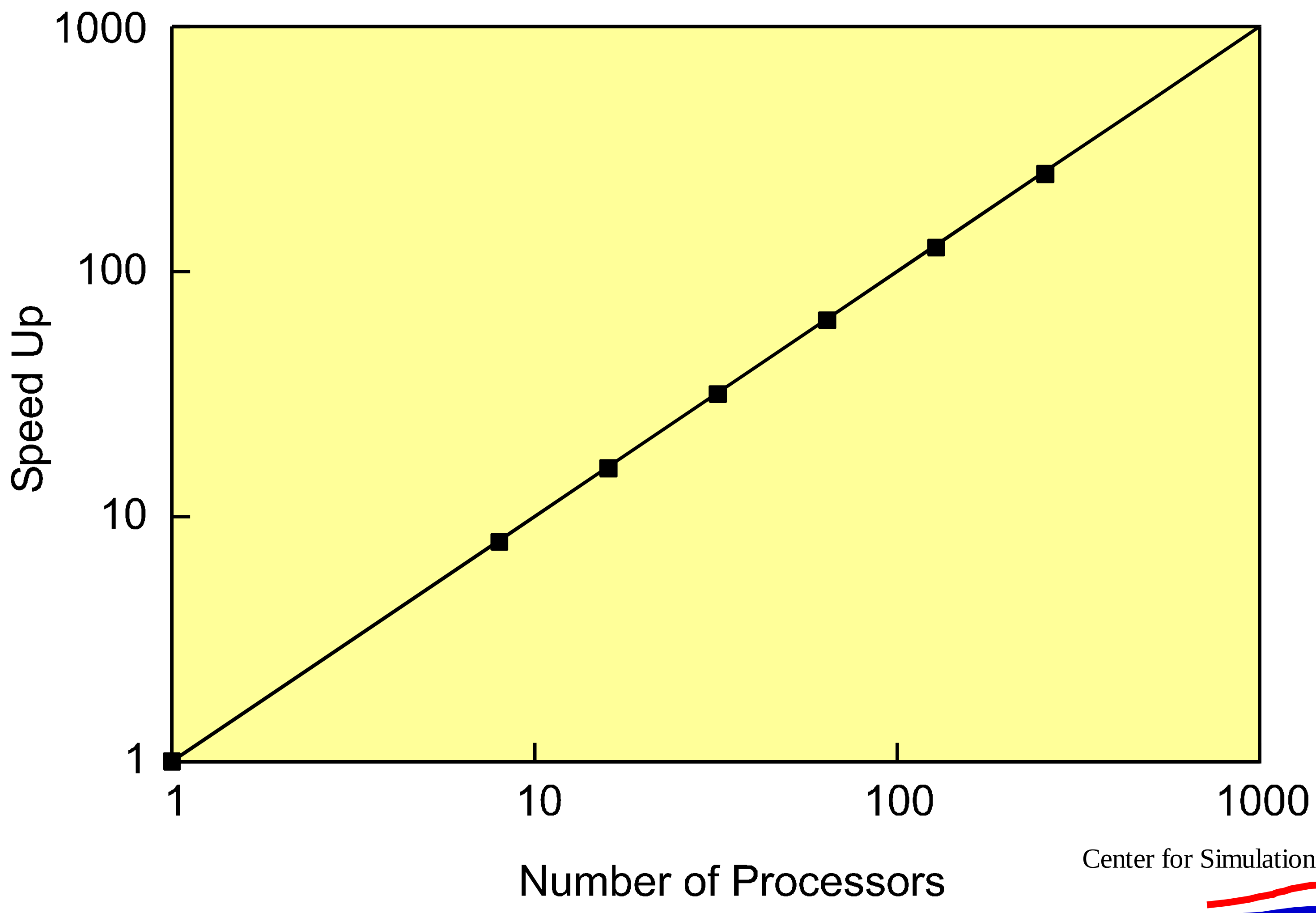
Origin 2000 Performance



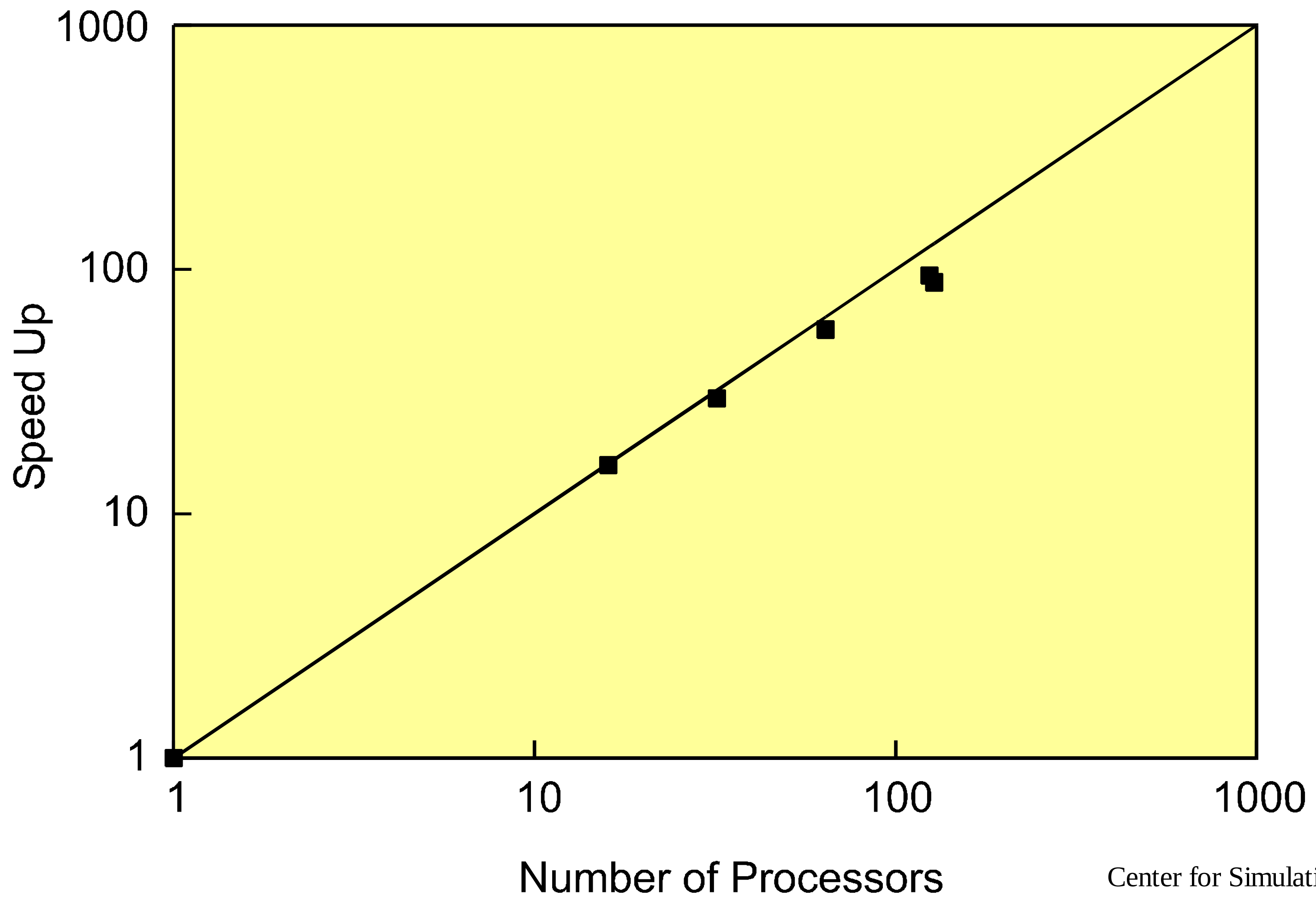
Coupled Solid Rocket Simulations

- **ROCSOLID**
  - **unstructured finite elements**
  - **implicit time integrator**
  - **multigrid equation solver**
- **ROCFLO**
  - **unsteady 3D, compressible Navier-Stokes equations on dynamic meshes**
  - **structured finite volumes**
  - **explicit time integrator**
- **Combustion model**
  - ***ap^n* used as interface regression rate**
- **Interface conditions**
  - **specify b.c.'s for each module**

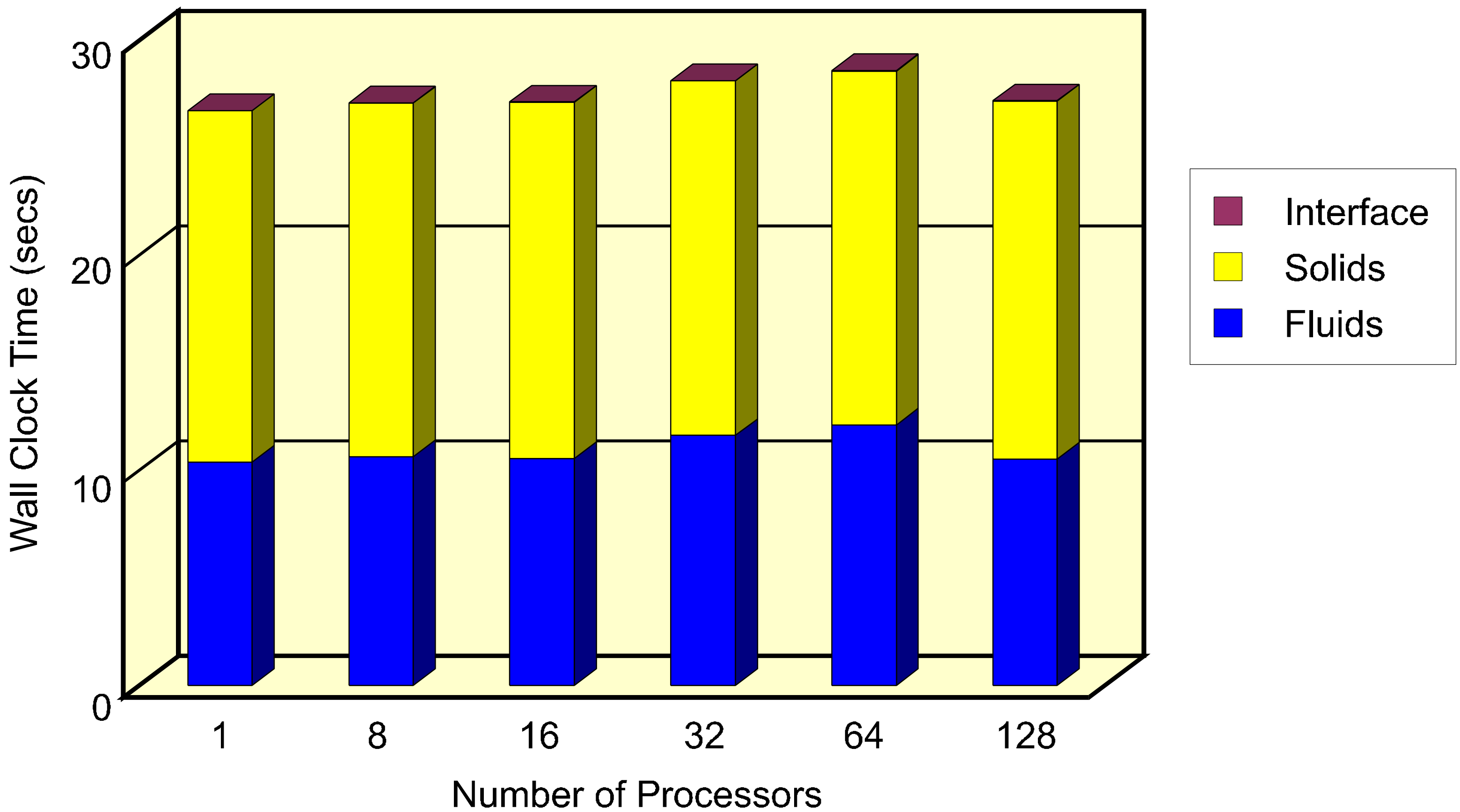
Cray T3E Scalability Results



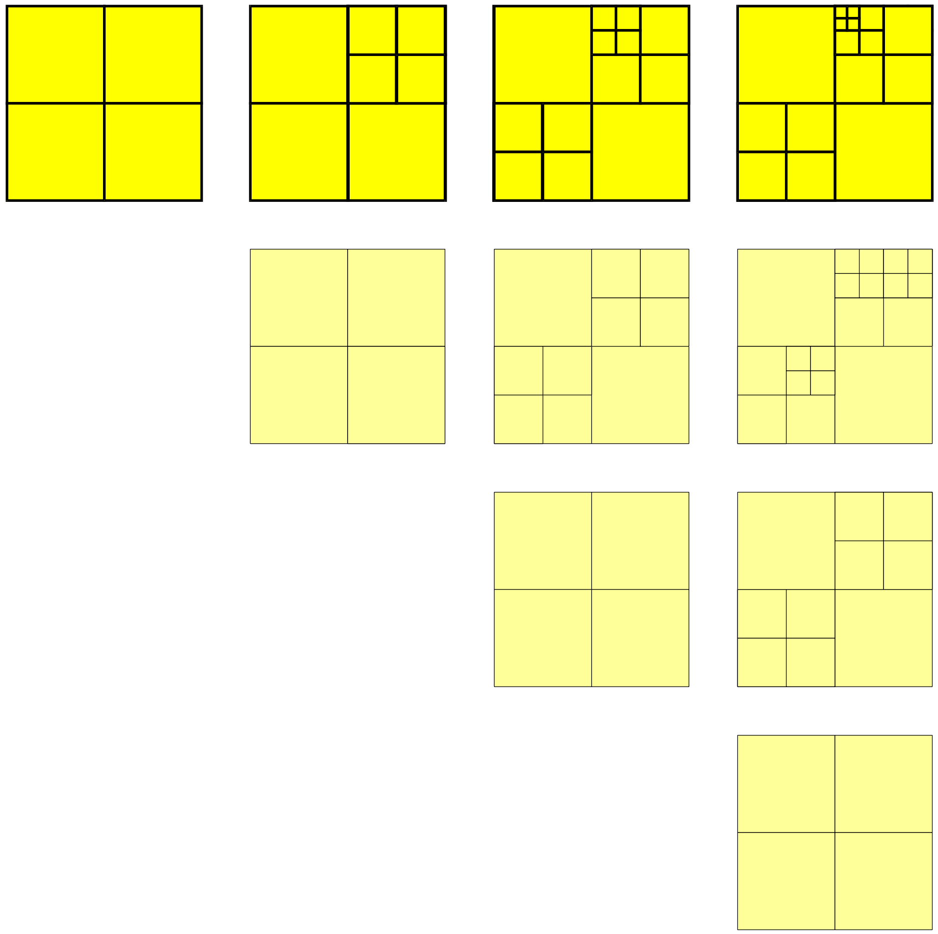
Origin 2000 Scalability Results

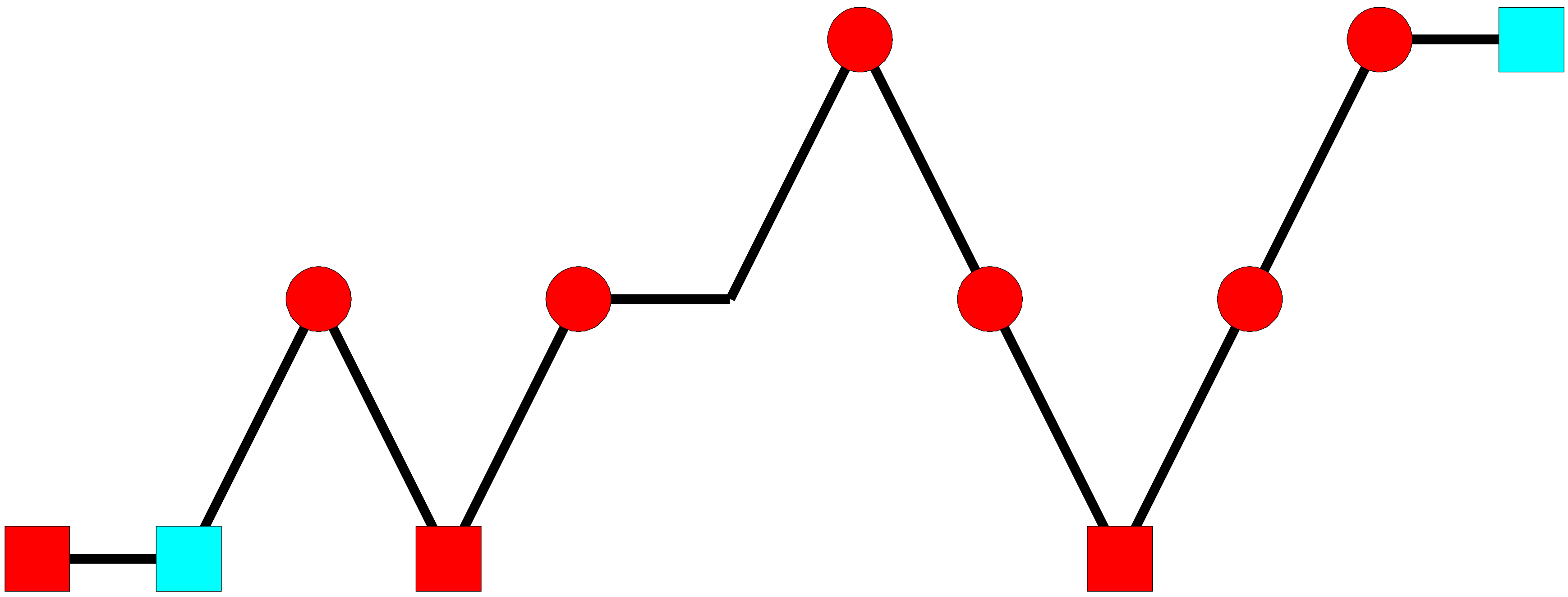


GEN1 T3E Time Requirements

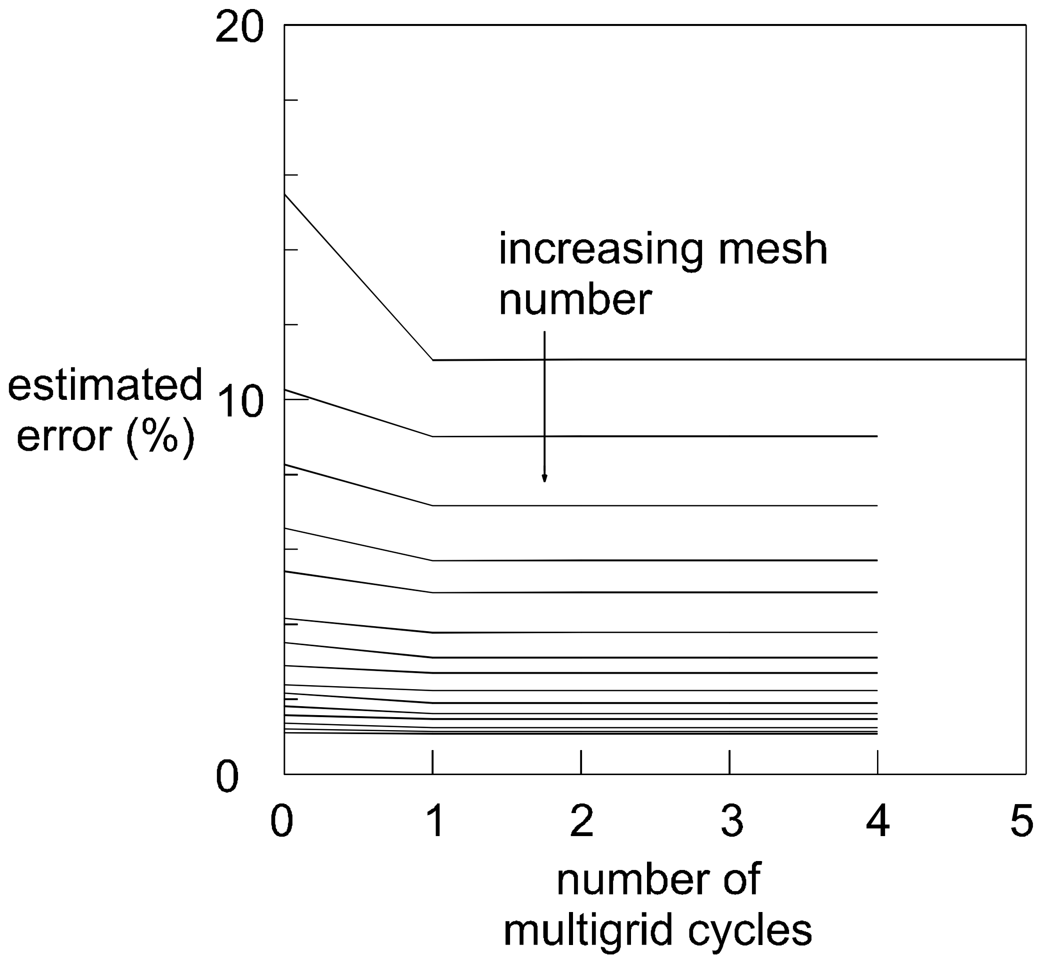


Adaptive Meshing and Multigrid

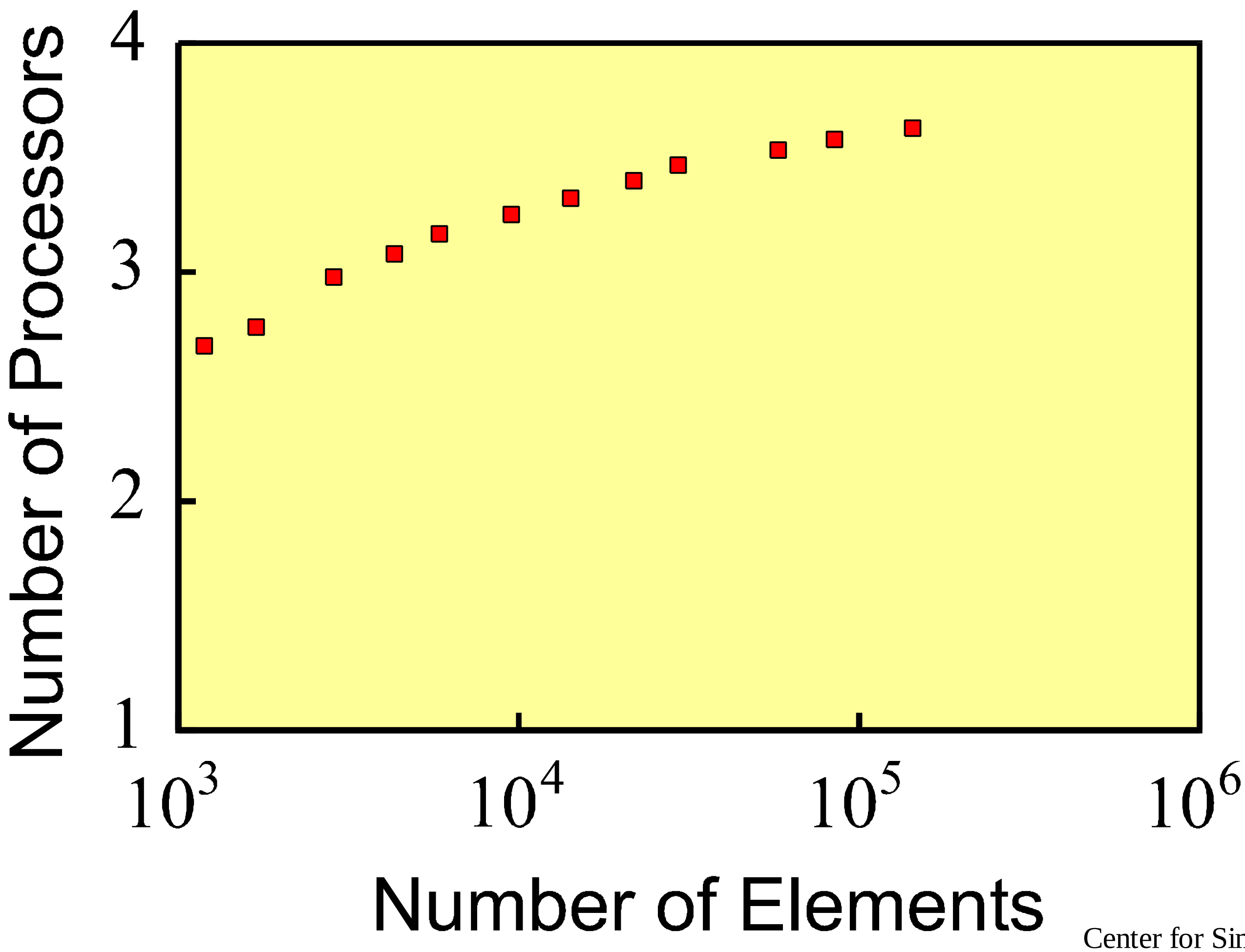




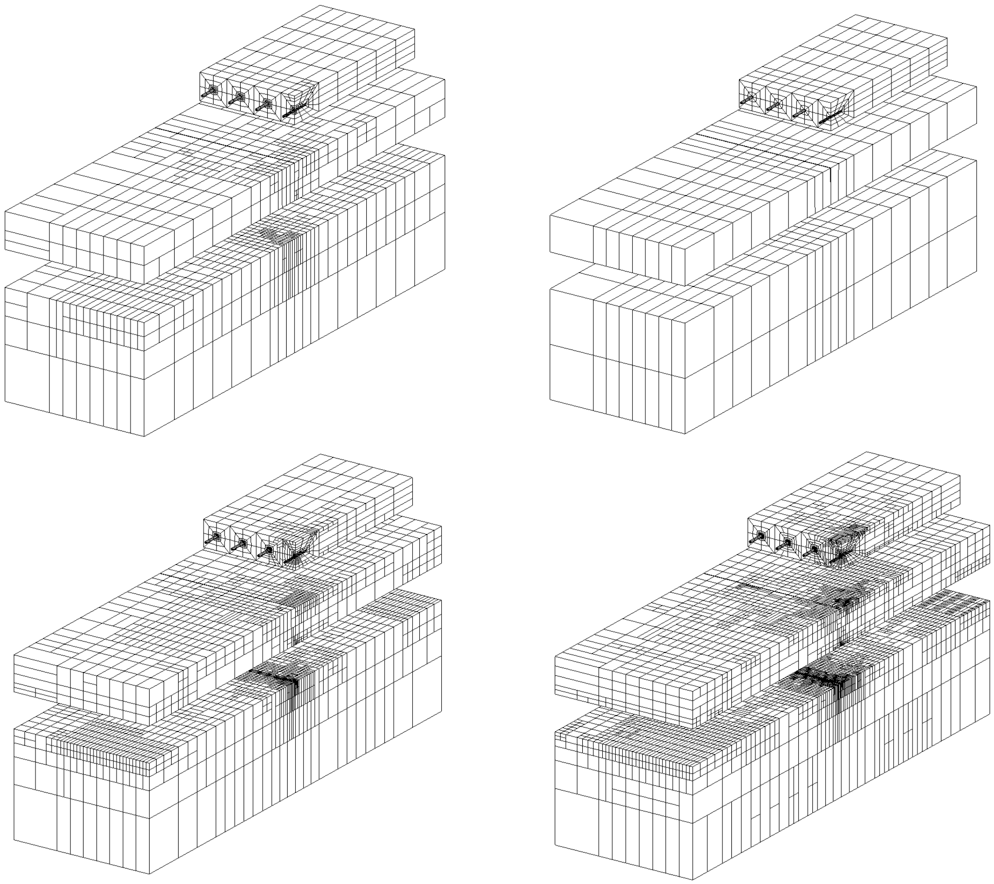
AMR-MG Convergence

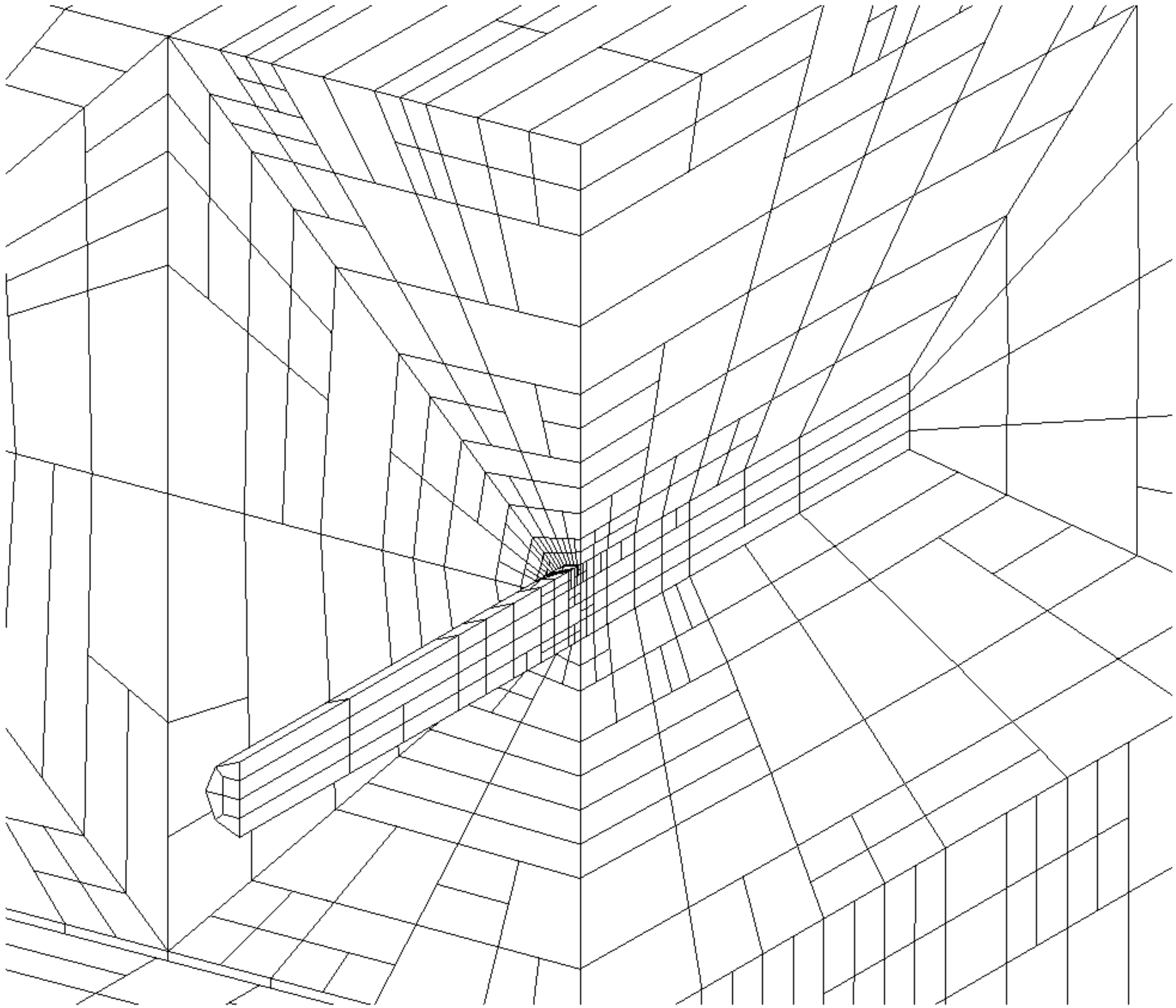


AMR-MG Parallel Speed-Up



AMR-MG Meshes





Future Plans

- **Shells using enhanced assumed strain solid elements**
- **Advanced material models**
- **Scalable nonsymmetric solvers for ALE**
- **Parallel contact algorithms**
- **Adaptive mesh refinement and shared memory parallelism**
- **Integration into Charm++ environment for AMR-MG in a distributed memory environment**

Scalable Multigrid Methods Using Fortran90 and MPI

Dennis Parsons and Ali Namazifard

Department of Civil Engineering

University of Illinois

December 1, 1999

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Center for Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign

ASCI
Accelerated Strategic Computing Initiative
Los Alamos
Sandia
Lawrence Livermore